# What’s New in Core Data

## Session 114

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

# What’s New in Core Data

**Adam Swift and Melissa Turner**
Core Data Engineering

# Introduction

- Snow Leopard features
- Accelerate development
- New performance options
- Best practices

# Roadmap

- Migration
- Spotlight integration
- Fetch performance
- Grand Central Dispatch
- Tips and tricks

# What You'll Learn

- Harness powerful new features
- Reduce, simplify code
- Less memory
- Faster performance

# Lightweight Migration

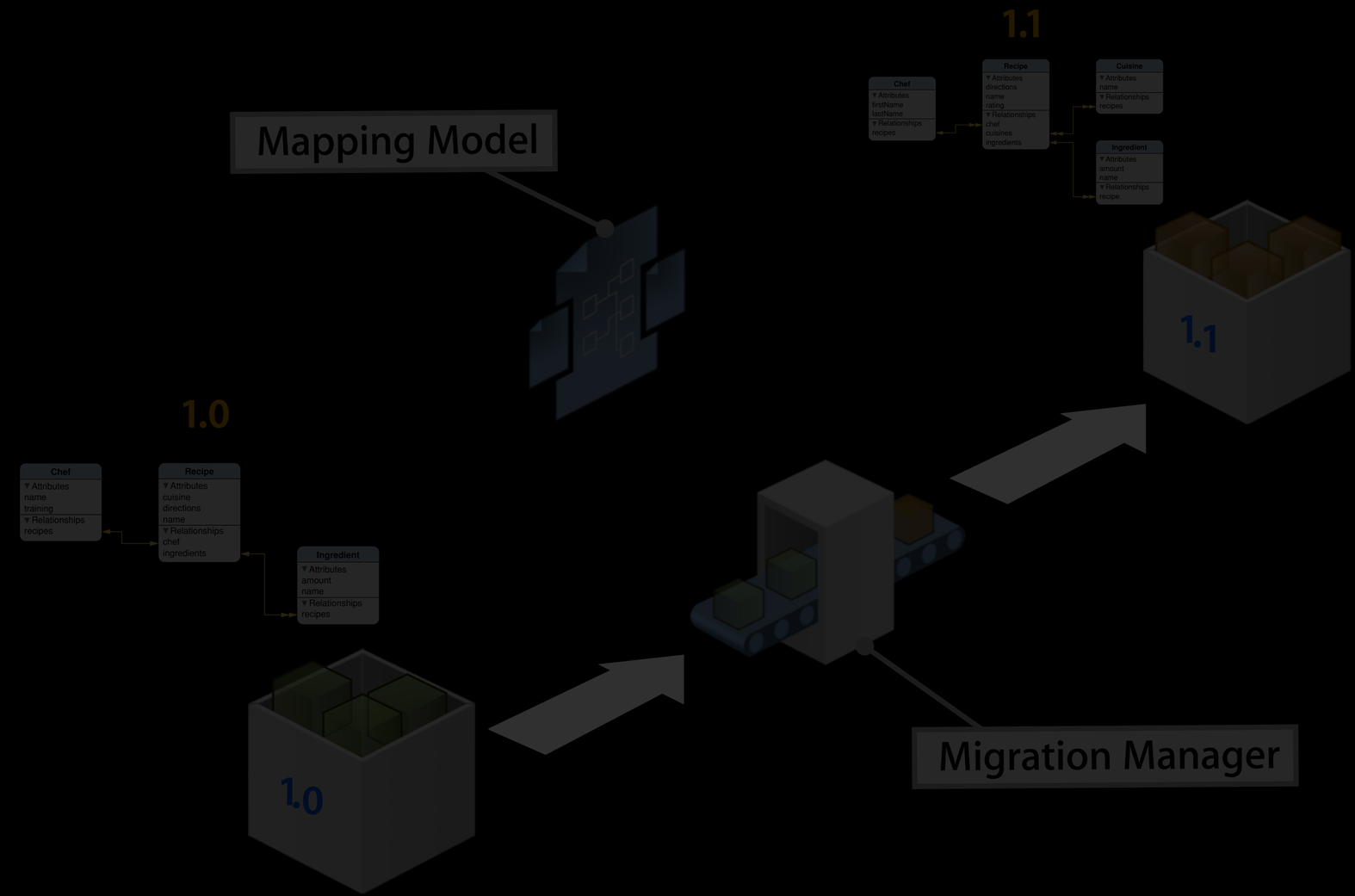
# Data Model Evolution

## No data model survives contact with reality

- Start with ideal model
  - Performance bottlenecks
  - Feature requests, user feedback
- Change your data model!

# The Migration Process

## Leopard schema migration



# Migration Basics

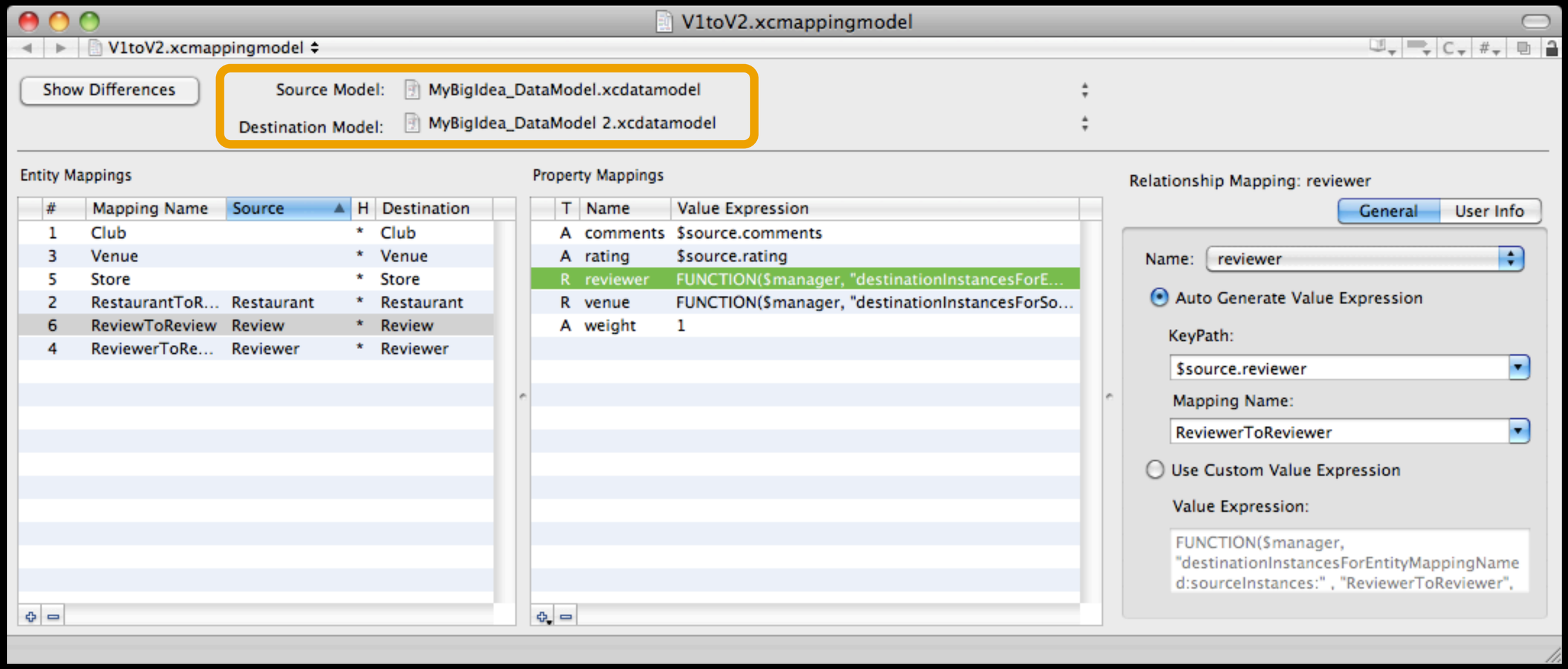
## Leopard schema migration

- Need compatible model to access data
- Create mapping model to migrate
  - Requires hand-tuning
  - Custom code for complex transformations

## Core Data Model Versioning and Data Migration Programming Guide

# Mapping Model

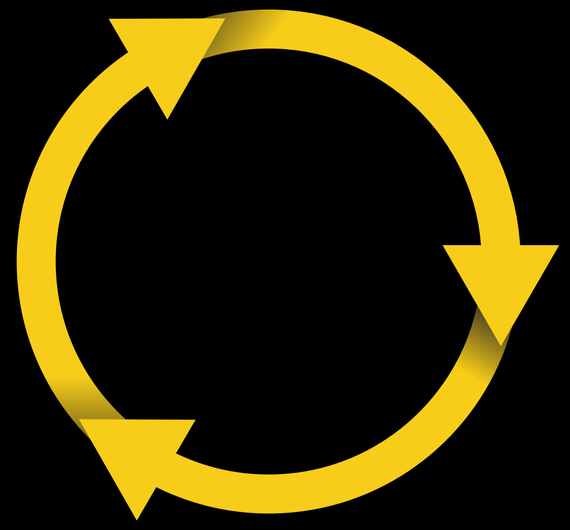
## Entity and property mappings



# The Development Cycle

## Edit, test, rinse, and repeat

- Incremental model changes
- Change is good
  - Performance
  - Features



# Lightweight Migration

## New automatic migration feature

- Simple data migration
- Easy setup
- Zero-code

# What Is Lightweight Migration?

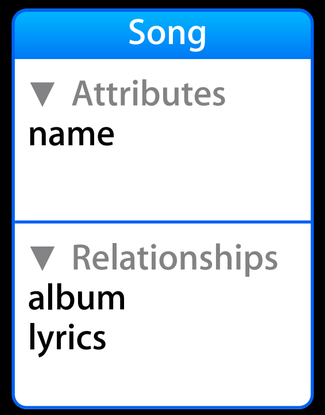
## Inferred data migration

- Automatically infer mapping model
- Handles unambiguous changes

# Inferable Changes—1

## Add, remove, or rename properties

### Version 1




### Version 2

# Inferable Changes—1

## Add, remove, or rename properties

### Version 1

**Song**

▼ Attributes
name

▼ Relationships
album
lyrics

### Version 2

**Song**

▼ Attributes
name
length

▼ Relationships
album
lyrics

# Inferable Changes—2

## Add, remove, or rename entities

### Version 1

**Musician**

▼ Attributes
name

▼ Relationships
albums

**Lyrics**

▼ Attributes
content

▼ Relationships
song

### Version 2

# Inferable Changes—2

## Add, remove, or rename entities

### Version 1

**Musician**

▼ Attributes
name

▼ Relationships
albums

**Lyrics**

▼ Attributes
content

▼ Relationships
song

### Version 2

**Musician**

▼ Attributes
name

▼ Relationships
albums
labels

**Label**

▼ Attributes
name

▼ Relationships
artists

**Lyrics**

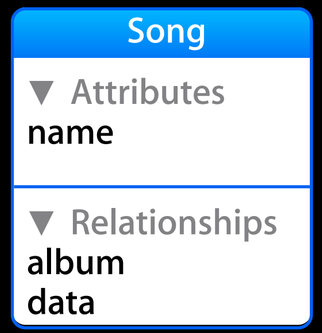
▼ Attributes
content

▼ Relationships
song

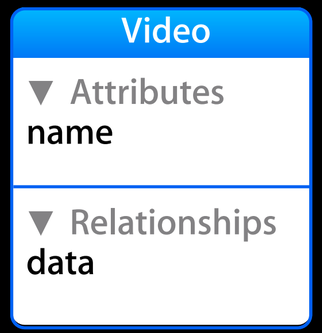
# Inferable Changes—3

## Change entity inheritance

Version 1




Version 2



# Inferable Changes—3

## Change entity inheritance

### Version 1

**Song**
- ▼ Attributes
  - name
- ▼ Relationships
  - album
  - data

**Video**
- ▼ Attributes
  - name
- ▼ Relationships
  - data

### Version 2

**Media**
- ▼ Attributes
- ▼ Relationships

**Song**
- ▼ Attributes
  - name
- ▼ Relationships
  - album
  - data

**Video**
- ▼ Attributes
  - name
- ▼ Relationships
  - data

# Inferable Changes—3

## Change entity inheritance

### Version 1

**Song**
- ▼ Attributes
  - name
- ▼ Relationships
  - album
  - data

**Video**
- ▼ Attributes
  - name
- ▼ Relationships
  - data

### Version 2

**Media**
- ▼ Attributes
- ▼ Relationships

**Song**
- ▼ Attributes
  - name
- ▼ Relationships
  - album
  - data

**Video**
- ▼ Attributes
  - name
- ▼ Relationships
  - data

# What You Need to Do

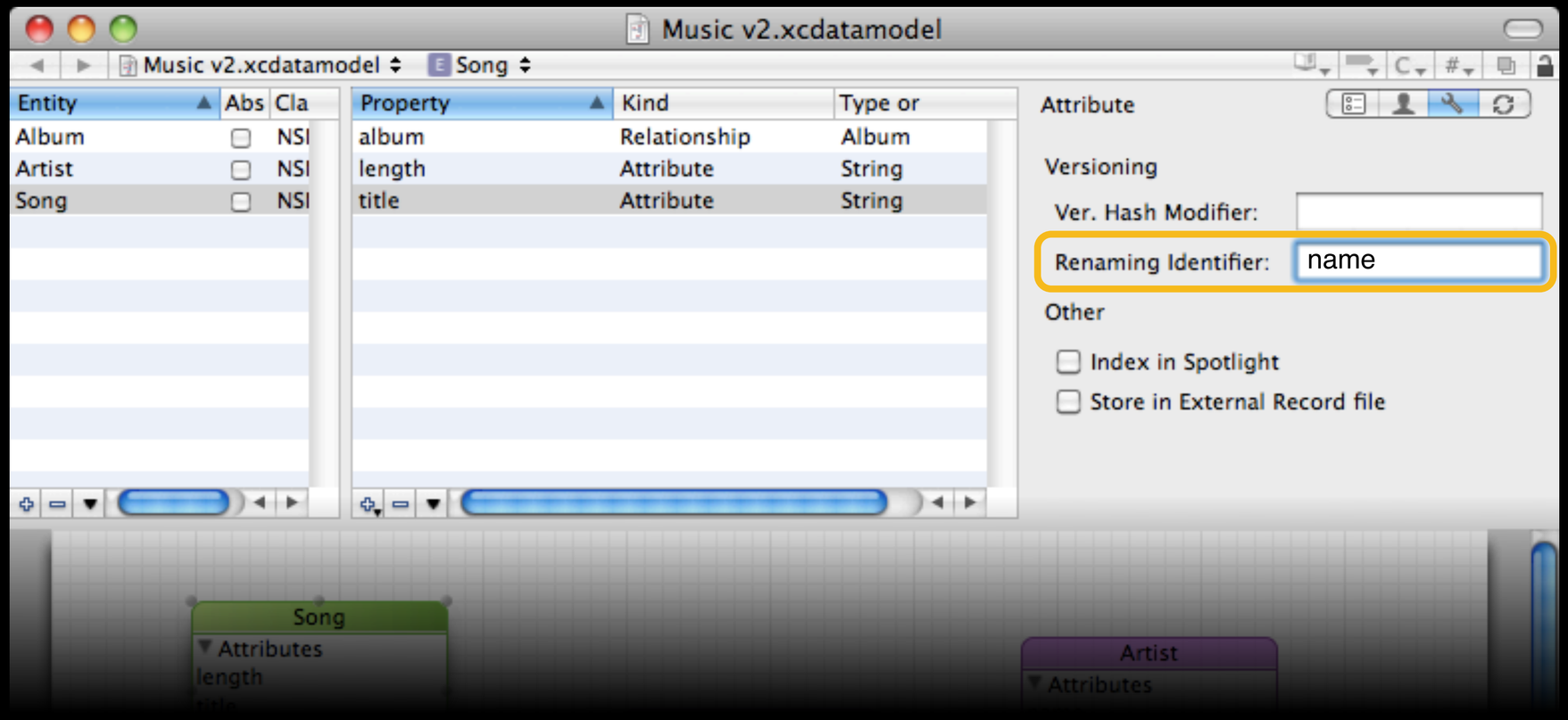
## Easy setup

- Keep your old model!
  - Xcode's **Design ➟ Data Model ➟ Add Model Version**
- Enable migration options
  - Infer mapping model automatically
  - Migrate persistent stores automatically

# Renaming

## Properties and entities

- Set the Renaming Identifier to the old name



# Fast and Efficient

## Minimal memory pressure

- SQLite stores
  - Migrates directly in the database
  - Uses SQL to alter tables, update rows
- Lightweight migration feature

# Debugging Lightweight Migration

## Desktop user-defaults to enable logging

- Migration logs

```
-com.apple.CoreData.MigrationDebug 1
```

- SQL logs

```
-com.apple.CoreData.SQLDebug 3
-com.apple.CoreData.SyntaxColoredLogging YES
```

Only on Mac OS

# Migration Summary

## Simple migration for simple changes

- Easy setup
- Very fast, minimal memory
- Plays nicely with custom mappings
  - Provide custom mapping model for complex changes

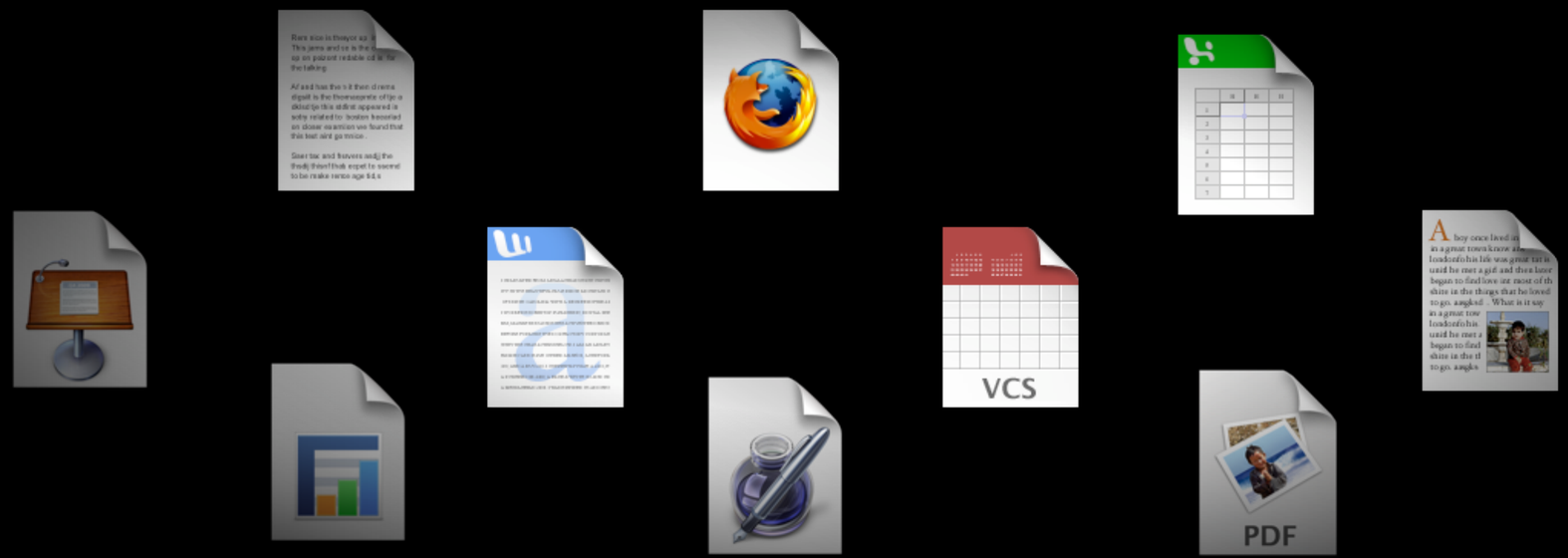
# Spotlight Search Integration

Only on Mac OS

# Spotlight Search

## Search for documents

- Document-based app Spotlight importer
  - Project template in Mac OS X 10.4
- Great for finding documents

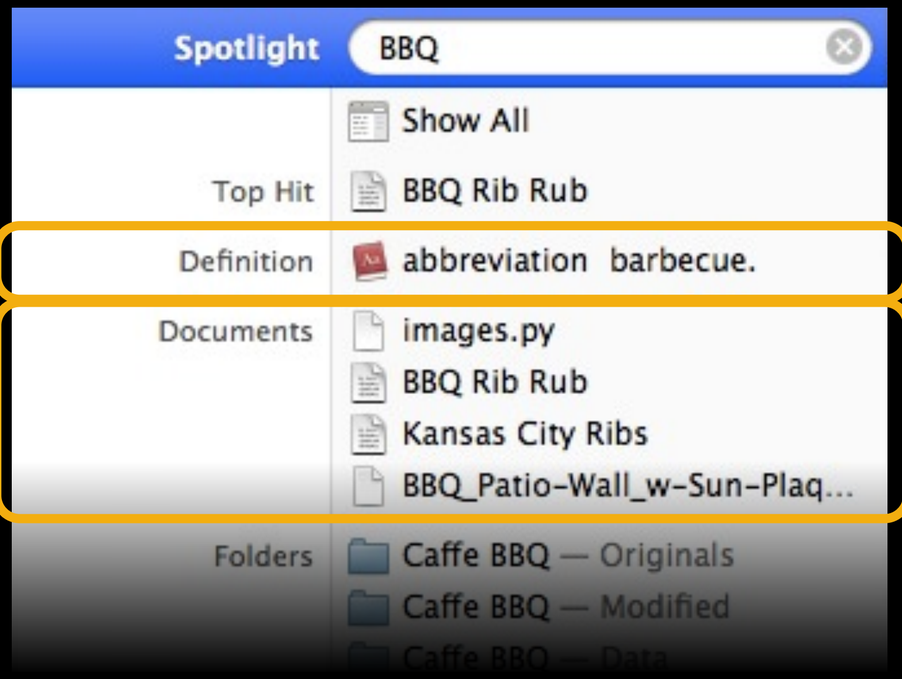



Only on Mac OS

# Spotlight Search

## Search for records

- Non-document apps
  - All records in one file
  - Fast lookup, efficient storage
- Want matching record…




Only on Mac OS

# Searching for Records

## An example

- Search for a name, “Malcolm”
- Results match individual records
  - Open in Address Book
  - Display matching contact!

Malcolm

Only on Mac OS

# New Spotlight Search Integration

## A better user experience

- For non-document apps
- Finding, opening individual records

# Core Data Spotlight Integration Programming Guide

Only on Mac OS

# Standard Spotlight API

## Importer creates index

- New spotlight importer template
- Imports from external record files

# External Record Files

## Managed by Core Data

- Each file identifies a managed object
- Property data still lives in the persistent store

Data Store

External Record Files

# Creating External Record Files

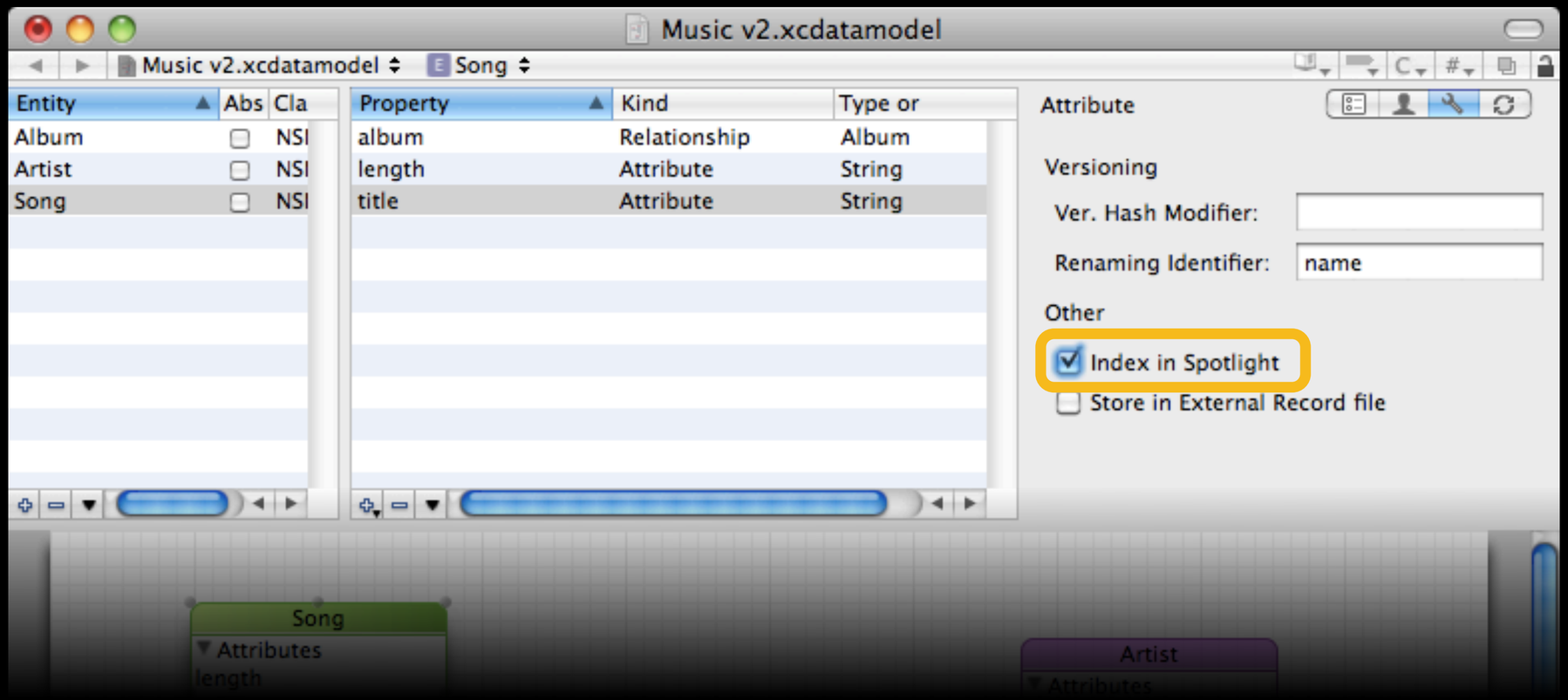
## Required settings

- Choose file extension
- Entities marked with indexed properties

# Index In Spotlight

## Xcode data modeling tool

- Enable the Index in Spotlight checkbox



# Synchronizing Records

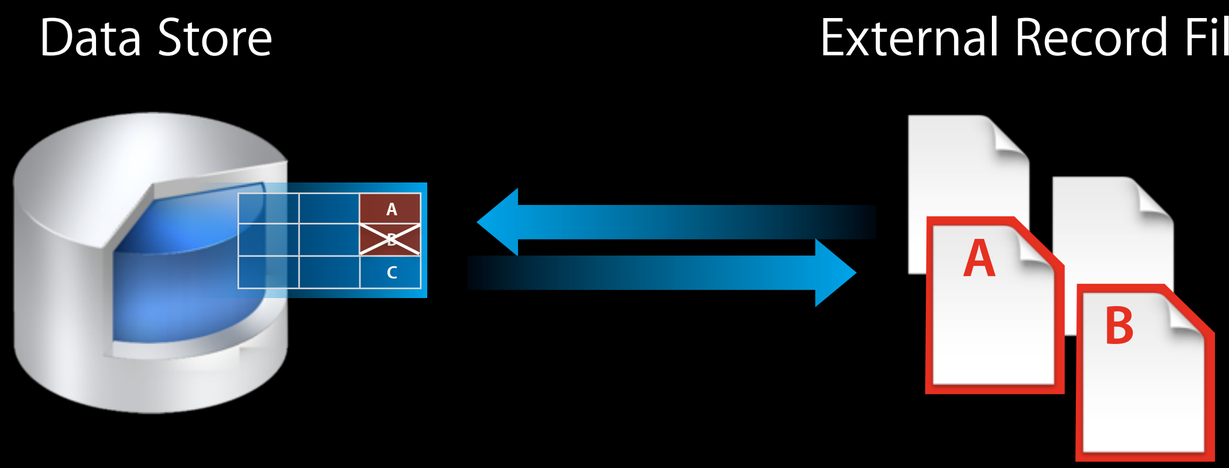
## We do the tricky part

- We sync external records with store
  - Save triggers sync

# Synchronizing Records

## We do the tricky part

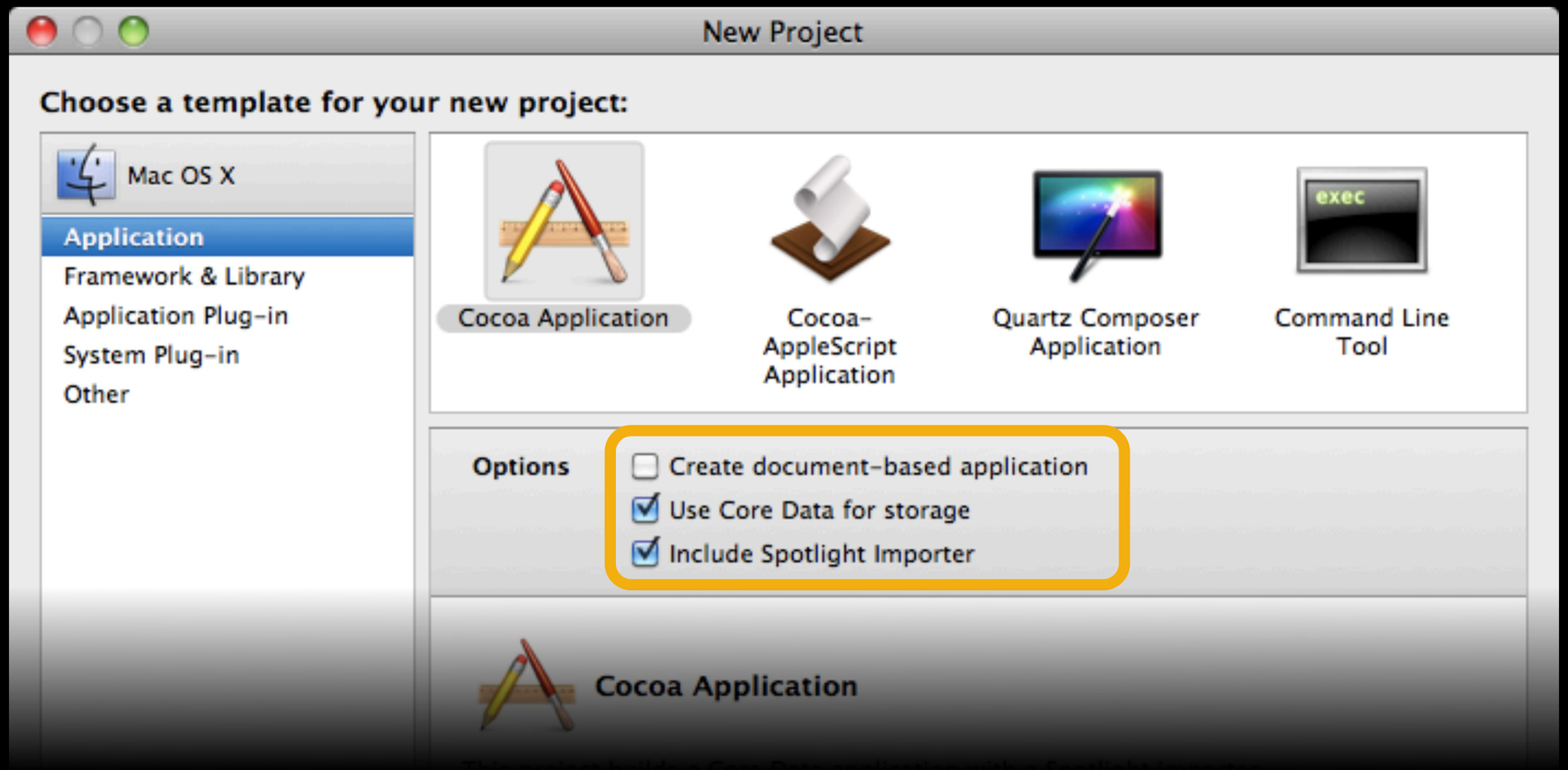
- We sync external records with store
  - Save triggers sync



# New Project Template

## Core Data Spotlight importer template

- New template for non-document Core Data apps
  - “Include Spotlight Importer”



# Spotlight Importer

## Customize the template code

- Lookup properties marked “Index in Spotlight”
- Add property data to the index
- …or any other data!  Everything in the store is accessible

# Displaying Search Results

## Open external records

- Register app to handle file extension
- Get the object ID from the file
- Fetch and display the record

# Spotlight Integration Summary

## Better search results

- Spotlight finds and opens individual records
- We'll maintain the record files
- You customize importer and display

Only on Mac OS

# Fetch Performance Options

# Fetch Performance

## Fundamentals

- Fetch only what you need
- How much on screen?
- Think “working set”

# Existing Fetch Options

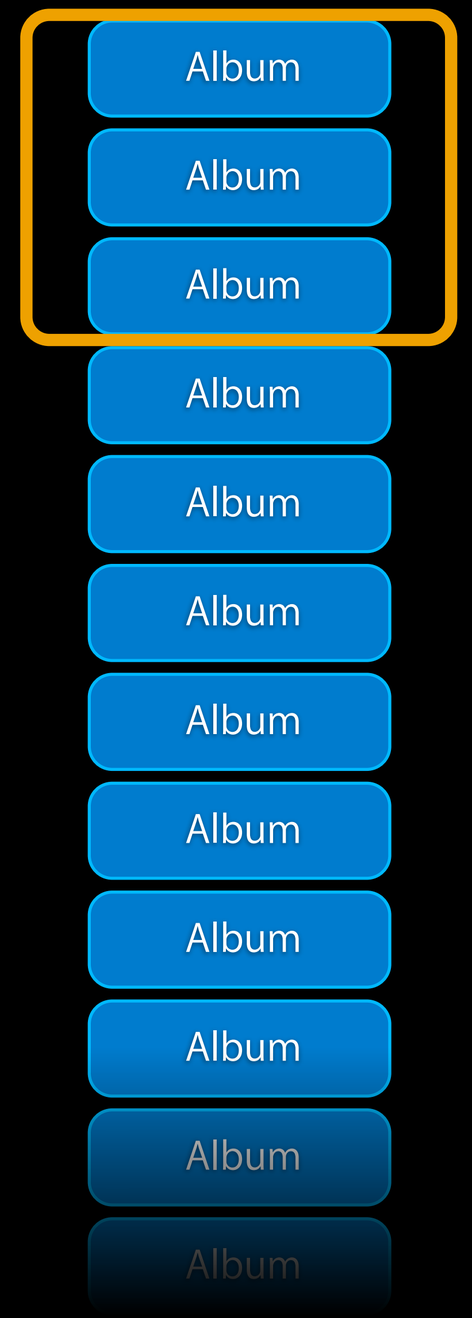
## Deal with large data set

- Fetch less data
- Managed objects
- Object IDs
- Count

# Vending a Large Data Set

## Interfacing with data consumers

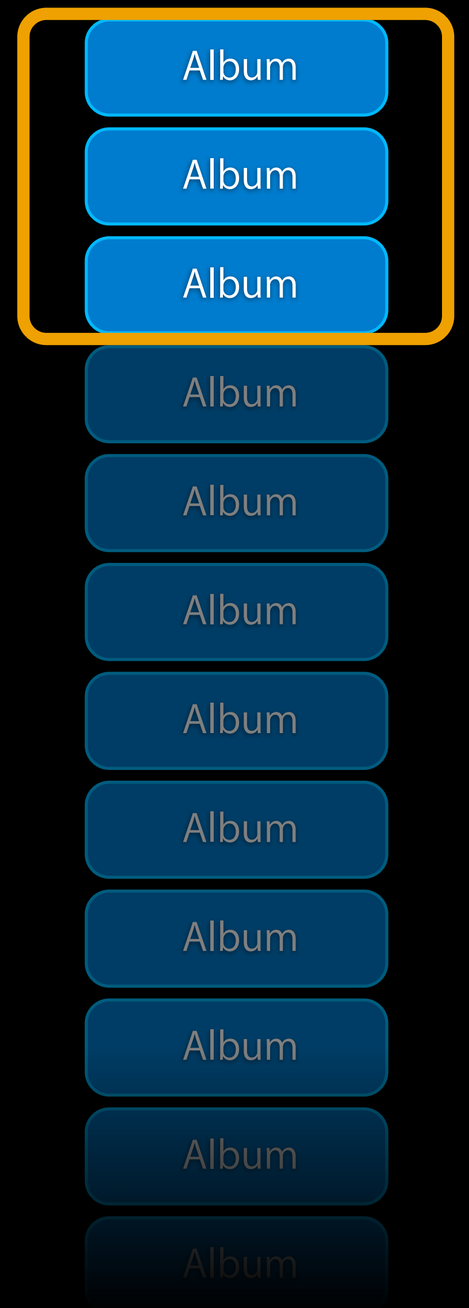
- UI wants whole array
- Access can be random
- Many records not displayed at all



# Fetching in Batches

## Sipping from a fire hose

- Provides random access to data set
- Limit memory use
- Amortize I/O



# Batch Fetching

## Fetch request configuration

- Choose batch size

  `setFetchBatchSize:`

- Returns auto-batching array proxy

# Large Data Set, Few Properties

## Partial faults

- Master/detail view
- Master view uses few properties
- Many undisplayed properties

# Lots of Rows, Little Data

## An example

| Title | Rating |
|---|---|
| National Hijinx | ★★★ |
| Little Fluffy Clouds | ★★★★ |
| Praise You | ★★ |
| Five | ★★★ |
| Belfast | ★★ |
| Nights Introlude | ★★★ |
| Little Green Rosetta | ★★★★ |
| Holiday In Berlin | ★★★★ |
| Ultraworld Nine | ★★★ |
| Biscuits | ★★★ |
| Desert Wind | ★★★★ |
| Around The World | ★★★★ |
| Battle Flag | ★★★ |
| Cowgirl | ★★★ |
| Sweet Lullaby | ★★★ |
| Tourance | ★★★★ |

# Partial Faults

## Only fetch what you'll use!

- Choose the properties

```
setPropertiesToFetch:
```

# Partial Faults

## Partially faulted objects

- Fetch managed objects
- Access can trigger full faulting
  - Non-fetched properties
  - Updates

# Includes Pending Changes

## Fetch request option

- Included by default

```
setIncludesPendingChanges:
```

# Fetch Includes Pending Changes

## Do you want them?

- Gives user a consistent view
- Deleted objects omitted
- Results affected by inserts and updates

# Bypass Pending Changes

## If you don't want them

- Just want saved data
- Perform query in SQLite only
- Lots of pending changes
  - In-memory evaluation can be expensive

# Fetch Options Summary

## Powerful new options

- Vend large data sets
  - In batches
  - Partial faults
- Bypass unsaved changes

# Advanced Fetch Techniques

**Melissa Turner**
Core Data Engineering

# Read-Only Core Data

## When partial faults are the solution to a different problem

- A different way to fetch only what you need
  - Aggregates
  - Distinct values
- Simple queries return simple values

# The Old Way of Getting an Aggregate

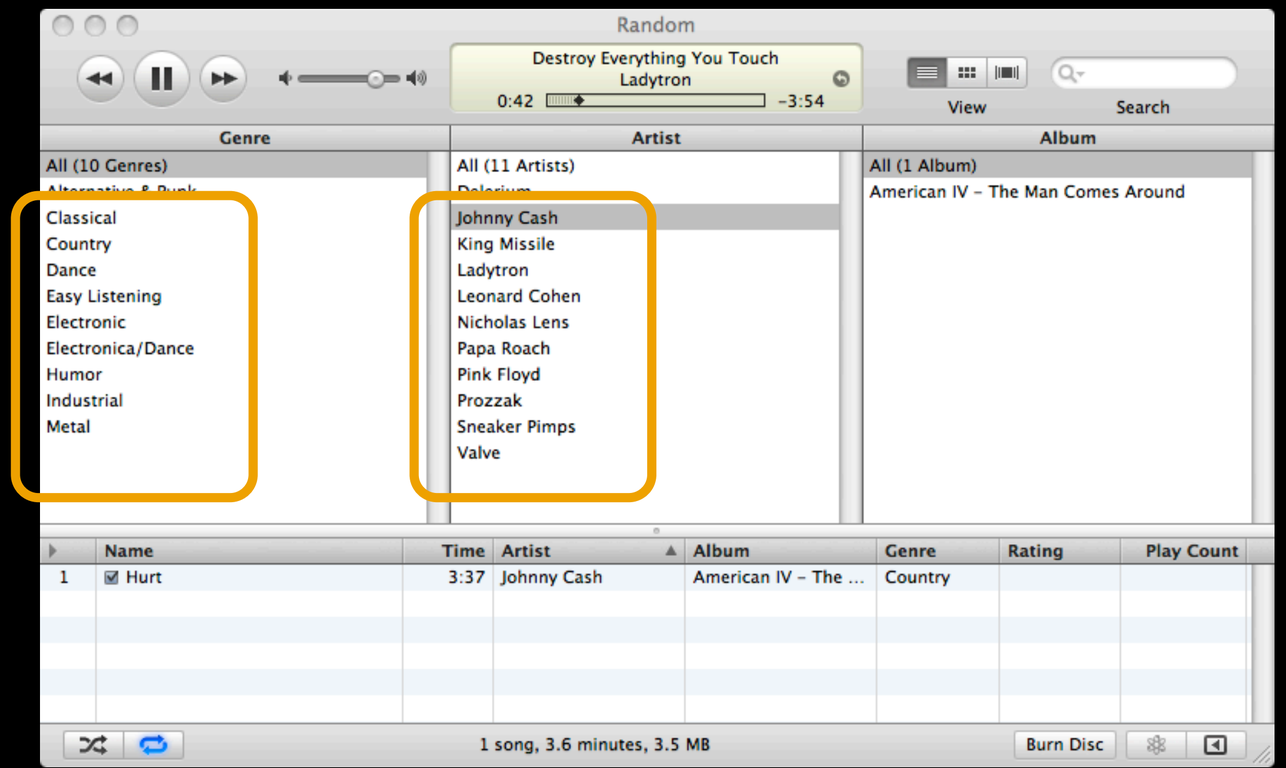
## Harder, worse, slower, less fun

- Load all of your objects
- Iterate through them, retrieving the pieces of data you need
- Do calculations
- Use the result

# The Old Way of Getting Distinct Values

## Harder, worse, slower, less fun

- Load all of your objects
- Iterate through them, collecting values in a set
- Use the set



# The New Way

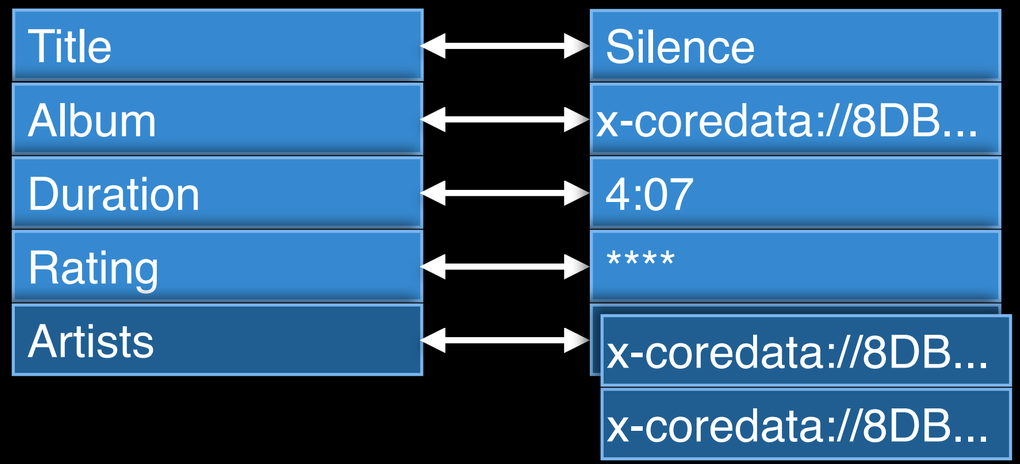
## Easier, better, faster, smarter

- Load only the exact data you want
  - Aggregate data in the database
  - Distinct values from the database
- Get results as a dictionary

# NSDictionaryResultType

## Easier, better, faster, smarter

- Returns flat dictionaries
  - Value classes
  - Object IDs
  - No to-many relationships
  - No collections



# Read-Only Data

## Easier, better, faster, smarter

- New class NSExpressionDescription
  - name
  - expressionResultType
  - expression
    - Supports most NSExpression types supported in predicates

For Duarte:

Please don't capitalize the sub bullets here; it's API and thus correct

# Read-Only Data

## Easier, better, faster, smarter

- Pass to NSFetchRequest
  - resultType
  - propertiesToFetch
  - returnsDistinctResults

For Duarte:

Please don't capitalize the sub bullets here; it's API and thus correct

# Example

## Fetching aggregates

```
NSExpression *expression = [NSExpression expressionForFunction:@”sum:”
arguments: [NSArray arrayWithObject: [NSExpression
expressionForKeyPath:@”duration”]]];
```

```
NSExpressionDescription *expressionDescription = [[[NSExpressionDescription
alloc] init] autorelease];
[expressionDescription setName: @”totalTime”];
[expressionDescription setExpression:[expression]];
[expressionDescription setExpressionResultType: NSDecimalAttributeType];
```

```
NSArray *properties = [NSArray arrayWithObject: expressionDescription];
[request setPropertiesToFetch: properties];
[request setResultType: NSDictionaryResultType];
```

# Example

## Fetching aggregates

```
{
 totalTime = 43108.08;
}
```

# Sample Code

Saving eyeballs and sanity one slide at a time

# Read-Only Core Data

# Example

## Fetching distinct values

```
NSExpression *expression = [NSExpression expressionForKeyPath:@”title”];
```

```
NSExpressionDescription *expressionDescription = [[[NSExpressionDescription
alloc] init] autorelease];
[expressionDescription setName: @”title”];
[expressionDescription setExpression:[expression]];
[expressionDescription setExpressionResultType: NSStringAttributeType];
```

```
NSArray *properties = [NSArray arrayWithObject: expressionDescription];
[request setPropertiesToFetch: properties];
[request setResultType: NSDictionaryResultType];
[request setReturnsDistinctResults: YES];
```

# Sample Code

Saving eyeballs and sanity one slide at a time

# Read-Only Core Data

# Expanded Predicate Support

# Expanded Predicate Support

## The other end of the SQL statement

- Support for KVC operators
  - @min/@max/@sum/@count

```
duration > duration.@average
```

# Expanded Predicate Support

## The other end of the SQL statement

- Support for functions
  - +, -, *, /, %
  - min:, max:, avg:, sum:
  - uppercase:, lowercase:
  - More—see NSExpression.h

```
lowercase:(title) CONTAINS ‘happy’
```

# Expanded Predicate Support

## The other end of the SQL statement

- New string comparison flags in parser
  - [n]ormalized
  - [l]ocale sensitive

```
title BEGINSWITH[n] ‘only’
```

# Expanded Predicate Support

## The other end of the SQL statement

- String comparison flags now work on equality/inequality operators

```
title <=[n] ‘only happy when it rains’
```

# Does It Make a Difference?

# Miscellaneous APIs

# New NSManagedObject APIs

- prepareForDeletion
- awakeFromSnapshotEvents:

# New NSManagedObjectContext APIs

- contextShouldIgnoreUnmodeledPropertyChanges
- existingObjectWithID:

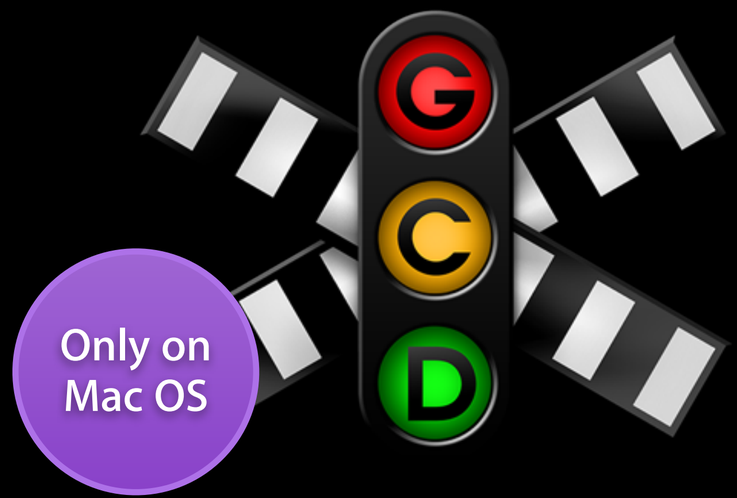
# Integrating with Other Mac OS X 10.6 Technologies

should this say "Snow Leopard Technologies?

# Grand Central Dispatch

## Running with scissors made easier

- System-level frameworks for making concurrent programming easier
  - NSOperation, libdispatch
- Serial or concurrent queues available
- Submit blocks for execution



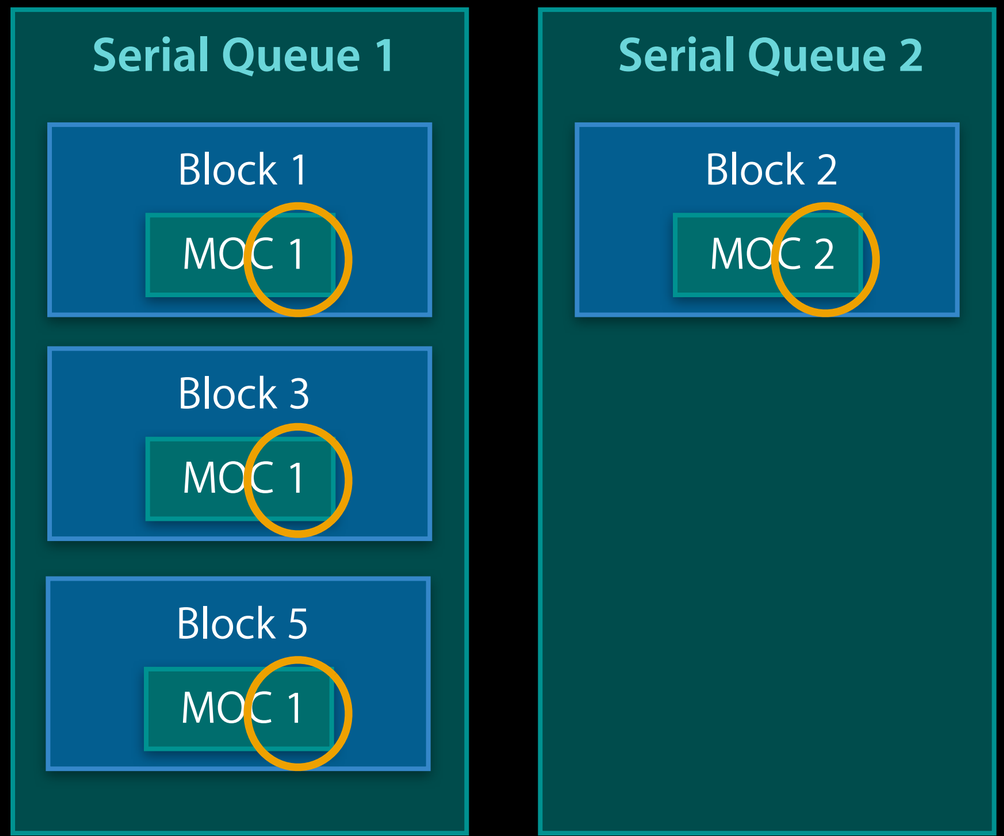
# Grand Central Dispatch

## Because this stuff is tricky

- CoreData Approved™ technique
  - Thread confinement
    - Threaded code: Each thread gets own MOC
    - GCD: Anything that might execute concurrently gets own MOC
  - Pass object IDs between threads

# For Visual Learners

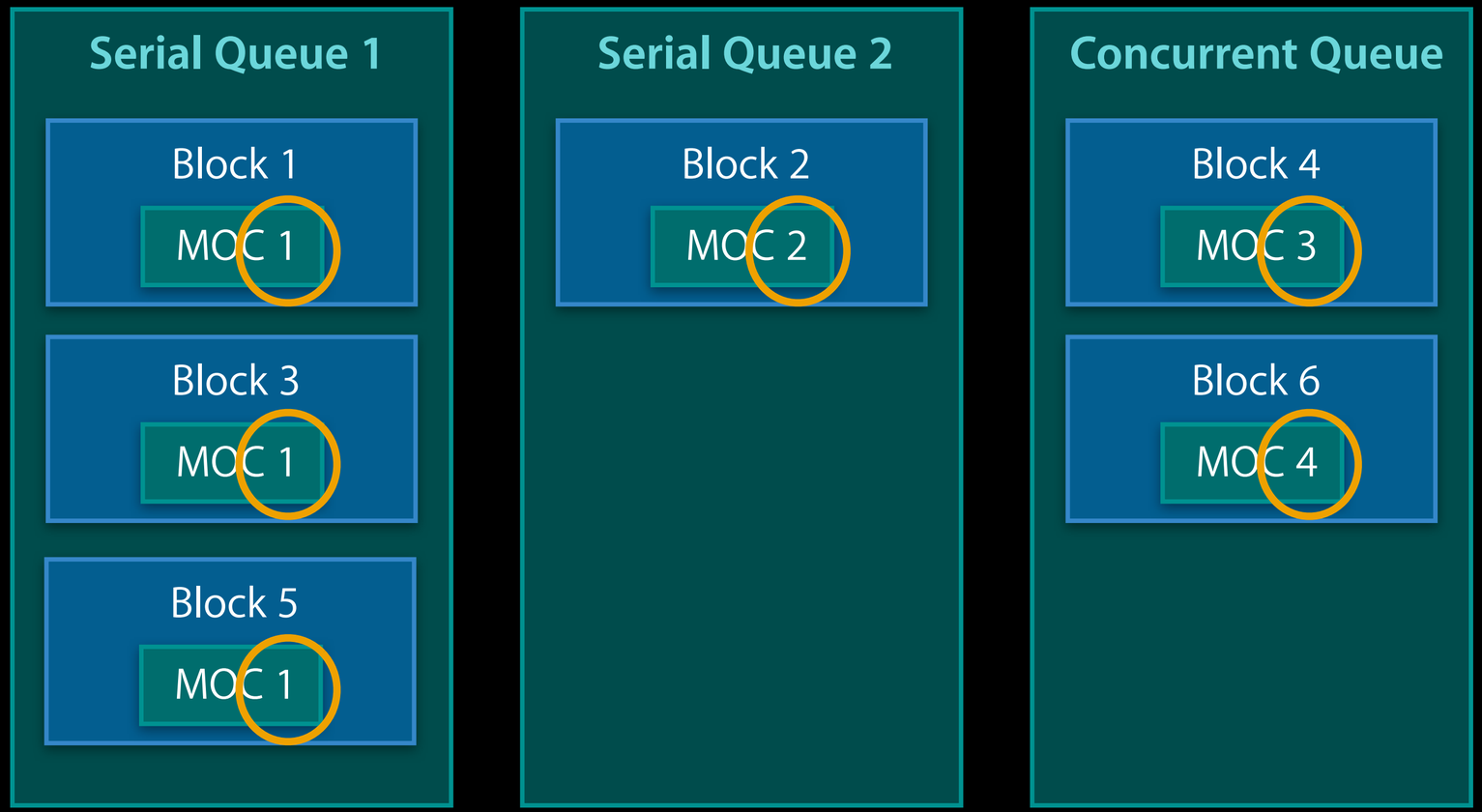
## Also because there can never be too many pictures




Only on Mac OS

# For Visual Learners

## Also because there can never be too many pictures




Only on Mac OS

# Debugging Threading Issues

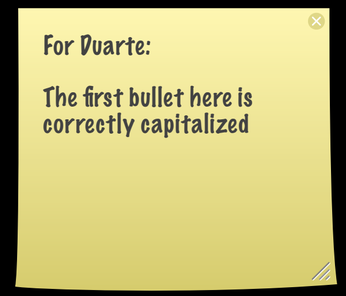
## Because this stuff is tricky

- Requires debug version of framework downloadable from ADC
  - -com.apple.CoreData.ThreadingDebug 3

Only on Mac OS

# Instant Off

- hasChanges is now observable
- Don’t change the object graph (we mean it)
  - performSelector:withObject:afterDelay: 0.0



# Summary

- Snow Leopard features
- Best practices
  - Core Data
  - Interacting with other technologies
- Accelerate development

# More Information

**Michael Jurewitz**
Application Technologies Evangelist
jurewitz@apple.com

**Documentation**
Starting point
http://developer.apple.com/documentation/Cocoa/Reference/CoreData_ObjC/index.html

# Got an Issue?

http://bugreport.apple.com

# Related Sessions

| | |
|---|---|
| Introduction to Core Data on iPhone | Marina<br>Tuesday 5:00PM |
| Migrating Your Application to Grand Central Dispatch | Pacific Heights<br>Thursday 2:00PM |

# Labs

| | |
|---|---|
| Core Data on iPhone Lab | iPhone Lab D<br>Wednesday 2:00PM |
| Migrating from SQLite to Core Data Tutorial Lab | OS X Lab C<br>Thursday 12:00PM |
| Core Data Lab | Snow Leopard Lab<br>Friday 9:00AM |

# Q&A

## WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09